REMARKS/ARGUMENTS

Status of Claims

Prior to entry of this amendment, the application included claims 1 – 6, 8 – 25, 37, and 38. Claims 1, 13, and 25 are amended; claims 37 and 38 are cancelled; and no claims are added. Hence, after entry of this Amendment, claims 1 – 6 and 8 – 25 stand pending for examination. Applicant respectfully requests reconsideration of this application in light of the amendments.

Claims 1 – 6, 8 – 25, 37, and 38 are rejected under 35 U.S.C. 112, first paragraph, as failing to comply with the enablement requirement. Also, claims 1 – 6, 8 – 25, 37, and 38 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over the cited portions of U.S. Patent Publication No. 2002/0129129 to Bloch et al. ("Bloch"), and further in view of the cited portions of U.S. Patent No. 6,874,143 to Murray et al. ("Murray"). Applicant traverses at least for the reasons below.

Rejections Under 35 U.S.C. § 112

Claims 1 – 6, 8 – 25, 37, and 38 are rejected under 35 U.S.C. 112, first paragraph, as failing to comply with the enablement requirement. Examiner contends, correctly, that it would make little sense to conflate the notions of asynchronous data transfer (as used in ATM networks) with the "executing … asynchronously" recitation of the independent claims. Office Action, p. 2 – 3. Indeed, this was not Applicant's intent in using the term "asynchronously," and the independent claims are currently amended to remove the term to more clearly recite their intended scope.

Specifically, amended claims 1, 13, and 25 recite "executing said application on said runtime environment independent from said program and independent from the program interacting with the server." Support for this amendment can be found throughout the specification. For example, pages 3, 7, and 9 all discuss embodiments of the invention being capable of running without a network connection (e.g., by storing data locally). This may be in contrast to many typical thin client approaches that may require constant, or at least periodic, client-server data transfers while the client is running. In Bloch, for instance, the AVM

apparently constructs virtual applications "on the fly," or "dynamically," by maintaining contact with web servers and databases (see, e.g., paragraphs [0032], [0034], [0037], [0038], [0047], [0058], [0060], etc.).

Applicant submits that the § 112 rejections to claims 1, 13, and 25, and to claims 2 – 5, 8 – 12, and 14 – 24 that depend therefrom, are moot in light of the amendments to the independent claims. Further, the § 112 rejections to claims 37 and 38 are moot in light of the cancellation of those claims. As such, Applicant respectfully requests withdrawal of the § 112 rejections to claims 1 – 6, 8 – 25, 37, and 38.

Rejections Under 35 U.S.C. § 103(a)

Independent claims 1, 13, and 25 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over Bloch, and further in view of Murray. To establish a *prima facie* case of obviousness, the cited references, combined with the knowledge of those of ordinary skill in the art, must teach or suggest all the claim limitations. Applicant respectfully traverses this rejection at least because the combination of the cited references with ordinary knowledge in the art fails to teach or suggest all the recitations of claims 1, 13, and 25.

Specifically, Bloch and Murray do not teach or suggest (1) "[executing said application] independent from the program interacting with the server," and (2) "executing said application on said runtime environment independent from said program," as recited in the independent claims. Regarding the first missing recitation, as discussed above, the AVM in Bloch apparently constructs virtual applications "on the fly," or "dynamically," by maintaining contact with web servers and databases (see, e.g., paragraphs [0032], [0034], [0037], [0038], [0047], [0058], [0060], etc.). As such, the application is executed dependently, not independently, on network interactions between the AVM and the server.

Regarding the second missing recitation, Bloch fails to teach a program that builds an application that is resident and executable on the client computer, separate from the program that built it, as recited in the claims. Bloch apparently discloses a program (i.e., an Application Virtual Machine, or AVM) that "downloads one or more text files from the server, retrieves program logic from each of the downloaded files, and assembles the retrieved program logic into a functioning, graphical application in temporary memory" (*see* Bloch, Abstract

(emphasis added)). Thus, essentially, Bloch seems to teach a program that only temporarily invokes an application, where the application is executable only within the program and runtime (the AVM) that invoked it (i.e., the application is spawned in RAM as a part of the AVM).

For example, according to Bloch, a user may run an application by accessing the AVM, waiting for the AVM to assemble the application in temporary memory from server-level application and data files, and interacting with the application through the AVM's runtime environment. The application apparently becomes merely a temporary function of the AVM, which may not be executed outside the AVM. Further, if the AVM in Bloch is deleted, the spawned "application" could no longer be accessed (i.e., the application could not be re-assembled for execution without reloading the AVM). This is different from using an agent program to compile an application so the application may be executable independent from the agent program in a runtime environment independent from the agent program. Unlike in Bloch, if the agent program is removed, the independent application may still be retained and executed in a wholly independent runtime environment.

Many portions of Bloch seem to support this understanding of Bloch as generally teaching that the application execution is performed by, and completely dependent on, the AVM. For instance, paragraph [0028] of Bloch describes the AVM as providing "a method for deploying and executing Extensible Markup Language applications . . ." Paragraphs [0030] – [0034] and [0044] include similar statements. Most striking, Paragraph [0047] states that "the AVM is used to execute 'Virtual Applications' that are assembled from files and programs residing on one or more computers on a network. Only AVM 221 is installed on a Client Device 10. Other files residing on Network Server(s) 131 are downloaded over a Network 121 to assemble the complete Virtual Application 20 that is then executed by the AVM 221 on the Client Device 10." Paragraph [0047] goes on to note that "[t]he XML Files 144 and Image Files 145 specify the appearance and behavior of the assembled Virtual Application 20 during execution by the AVM 221 and in response to user interactions during such execution."

The portions of Bloch cited by the Office Action even further seem to support Applicant's reading of Bloch. For example, the Office Action points to paragraphs [0086] and [0087] as teaching an application that is executable independent from the program that created

the application. However, Paragraph [0086] describes tasklists that include client tasks, conditional tasks and host tasks, and Paragraph [0087] states that these "[t]asklists are executed by the AVM when an event occurs to which they are linked." A tasklist cannot be both executed by an AVM and executable independent from the AVM.

The Office Action also points to paragraph [0100]. This paragraph describes the interaction of the application and the AVM. For example, paragraph [0100] describes "[a]s the user interacts with components, the AVM 221 detects, in a platform dependent way, changes to the visual appearance of the component that reflect the entry of data, the selection of a choice, putting focus on a component, and the like (step 92)." The paragraph also describes how an AVM session is ended after function is executed by the system handler. If the function (or application) is truly independent of the AVM, why then is the AVM terminated at the end of the function? Thus, the application and AVM are dependent applications rather than independent applications.

The Office Action also points to paragraph [0102] as describing message boxes that are unique to the operating system. These message boxes occur in response to some action by the AVM. Thus, these message box tasks are dependent on the AVM and not an application created by the AVM. Moreover, these message boxes are, as noted by the Office Action, operating system procedures that are not an application created by the AVM from downloaded text files. The claims require that the same application that is created by the AVM execute independent from the AVM. Operating system procedures or calls have not been downloaded by the AVM.

The Office Action also points to paragraph [0103] as describing remote procedures. Remote procedures are procedures that occur remote from the client computer system and not at the client computer system. The claims require, in part, that a program (i.e., the AVM) create an application from text files on the client computer system and that the application is executable independent from the program. Simply put, remote calls are not executed on the client computer system. Moreover, these remote calls are not part of the application downloaded and created by the AVM.

The Office Action also points to paragraph [0105] of Bloch as referring to a database handler. This paragraph does not teach or suggest that the database handler is executable independent of the AVM. There is also nothing in Bloch to suggest that the database handler is created by the AVM from text files.

The Office Action also points to paragraph [0109] of Bloch as referring to "user interface . . . accessing any database . . . to test new software." Paragraph [0109] describes how the AVM is used to test software, assemble user interfaces, access any database, etc. Therefore, the AVM as described is not independent from anything. It is completely dependent with software testing, user interface assembling, and database accessing.

From these and other exemplary portions of Bloch, it is clear that the application is not executable independent from the AVM or independent of server interactions. For at least these reasons, Bloch fails to teach or suggest all the recitations of independent claims 1, 13, and 25. Further, the Office Action provides no suggestion that Murray or ordinary knowledge in the art includes any teachings to remedy these deficiencies of Bloch. As such, the combined teachings of the art, as cited by the Office Action, fail to establish a *prima facie* case of obviousness as to independent claims 1, 13, and 25.

Applicant respectfully submits that the specified limitations in independent claims 1, 13, and 25 are allowable for at least the foregoing reasons. Claims 2 – 6, 8 – 12 and 14 – 24each depend from these independent claims, and are believed allowable for at least the same reasons as given above. Applicant, therefore, respectfully requests that the §103(a) rejections to these claims be withdrawn.

CONCLUSION

In view of the foregoing, Applicant believes all claims now pending in this application are in condition for allowance. The issuance of a formal Notice of Allowance at an early date is respectfully requested.

If the Examiner believes a telephone conference would expedite prosecution of this application, please telephone the undersigned at 303-571-4000.

Respectfully submitted,

Date: October 16, 2008

Daniel J. Sherwinter
Reg. No. 61,751

TOWNSEND and TOWNSEND and CREW LLP
Two Embarcadero Center, Eighth Floor
San Francisco, CA 94111-3834
Tel: 303-571-4000
Fax: 415-576-0300
DJS/sk
61491709 v1